May 2019

A Simplified Manual of the JSBSim Open-Source Software FDM for Fixed-Wing UAV Applications

TECHNICAL REPORT

Oihane Cereceda

Faculty of Engineering and Applied Science
Memorial University of Newfoundland
St. John's, NL, Canada

Summary

Simulation packages provide a valuable framework or environment to study the interaction between aircraft, including Unmanned Aerial Vehicles (UAVs), and the existing air traffic in Near Mid-Air Collision (NMAC) scenarios. The described simulation package is based on the open-source JSBSim Flight Dynamics Model (FDM), which has been validated and tested in UAV computer models for 4D encounters and avoidance manoeuvres. The objective of this technical report is to provide a simplified version of the current package, including the minimum requirements for the design of a UAV in JSBSim, and to guide any modellers on the UAV computer design task. Introductory concepts and the dynamics behind this package will not be stated here.

This report begins with a brief introduction of JSBSim structure and simulation modes. The source code classes are introduced in Section 2 followed by the set instructions for the additional feature of the multiplayer mode used in 4D encounters. The report concludes with a UAV example case study.

Table of Contents

List of Figures

List of Acronyms

ATC	Air Traffic Control
CSV	Comma-Separated Values
DoF	Degrees of Freedom
FDM	Flight Dynamics Model
GNC	Guidance, Navigation, and Control
NL	Newfoundland and Labrador
NMAC	Near mind-Air Collision
R/C	Radio Control
RAVEN	Remote Aerial Vehicles for ENvironment-monitoring
UAV	Unmanned Aerial Vehicle

1. High-Level Simulation Structure

JSBSim [1] is an FDM package that consists of a series of classes integrated together to simulate an aircraft and its environment. The package is stable and ready to use from the command/shell window. The basic version also includes a large aircraft library and simple demos. The most remarkable work done on JSBSim are the motion base simulator at the University of Naples, Italy [2] and the human pilot math model with JSBSim as the 6-DoF simulation core, developed by the U.S. Department of Transportation [3].

However, JSBSim does not contain any visual environments or models associated with it and additional software –FlightGear [4]– is required if the performance needs to be observed.

JSBSim can be downloaded online [5] and more information can be found on its website [6]. Although the developers claim that it can be integrated into MATLAB/Simulink, the system is still in development and needs significant improvement.

1.1. Standalone Mode: Scripting

The standalone simulation mode of JSBSim only requires the source code, the set of engines, and aircraft. The package is in constant development with periodic releases; the source code is considered stable whereas new aircraft and other systems are uploaded to the repository after they are verified. If the user needs to compile and build the program, they should follow the instructions in the manual [1].

A JSBSim standalone simulation structure can be summarized by the following blocks (Figure 1):

- The **JSBSim source code (B1)** is formed by the full 6-DoF FDM, including the dynamics of the system and all the models that exchange properties with the aircraft or computer model, such as wind and atmosphere.
- The **script file (B2)** is expressed in .xml format and describes the tasks to be performed by the computer model.
- The **initialization file (B3)** includes the information related to the initial state of the aircraft. It can be called either from **B1** or **B2**.
- The **aircraft configuration file (B4)** contains all the parameters for a specific aircraft. The main file includes the metrics, mass and aerodynamics, among others. The propulsion system formed by the **engine (B4.1)** and **thrust (B4.2)** files are called from **B4**. **Extra files (B4.3)** can also be added depending on the final purpose of the

simulation. Examples include an autopilot, guidance system, or more specific information about elements such as sensors or a control system.

- The type of **output (B5)** is defined at the end of the aircraft configuration file (**B4**) and generated by the source code. The output can be generated through a series of datalogs or, become a visual performance using a complementary software such as Flight Gear or OpenEaagles [7].
- An **additional datalog (B6)** with specific parameters or a set of parameters to a particular package can be added as well.

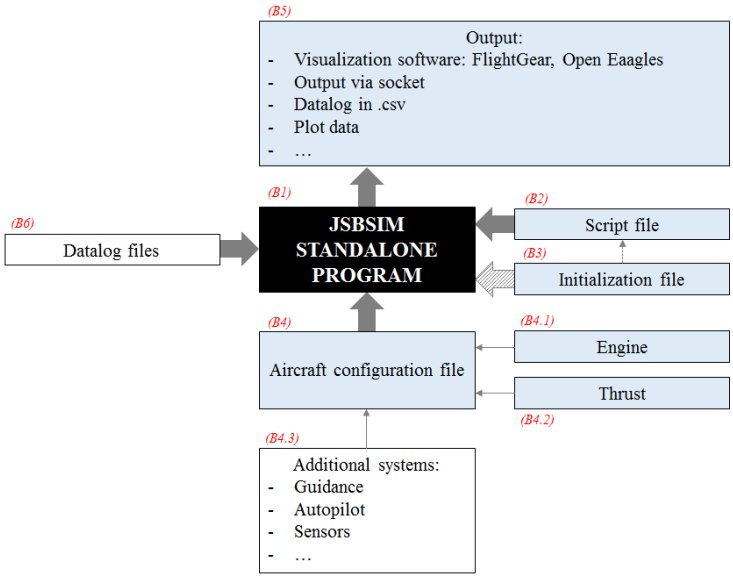


Figure 1. JSBSim standalone mode structure

Therefore, the minimum and most basic configuration consists of the source code (**B1**) with the aircraft configuration file, including the propulsion system (**B4**, **B4.1**, **B4.2**), the script files defining the task (**B2**), the initialization file (**B3**), and the output generation files or interface (**B5**). The corresponding blocks are highlighted in light blue in Figure 1. The remaining blocks are complementary depending on the task to be completed. A straightforward simulation can be run with the Debug command shown in Figure 2, indicating the minimum files and configuration/execution simulation parameters.

If run in batch mode, the complete simulation is executed in a line code (Figure 3). That batch file should include the basic and minimum command lines for the simulation as described below in the code box. This example eliminates files generated from previous simulations, runs the JSBSim package by indicating the correspondent script, moves the output file and plots the results in Gnuplot [8].

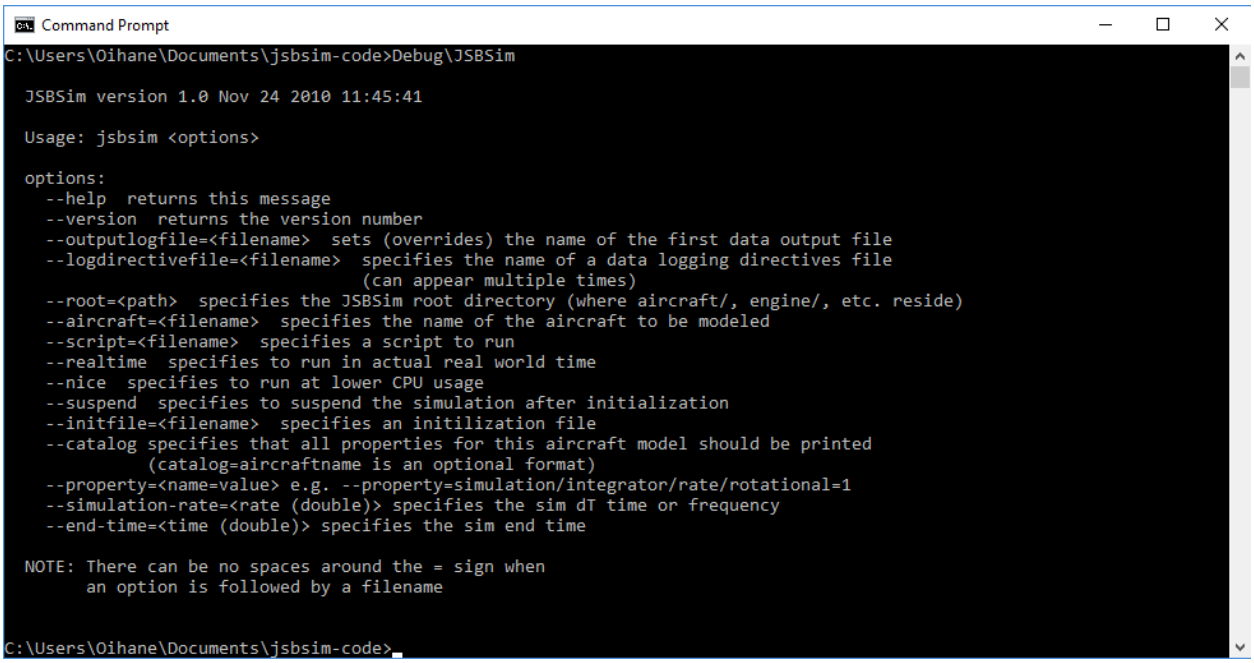


Figure 2. JSBSim program command line and options

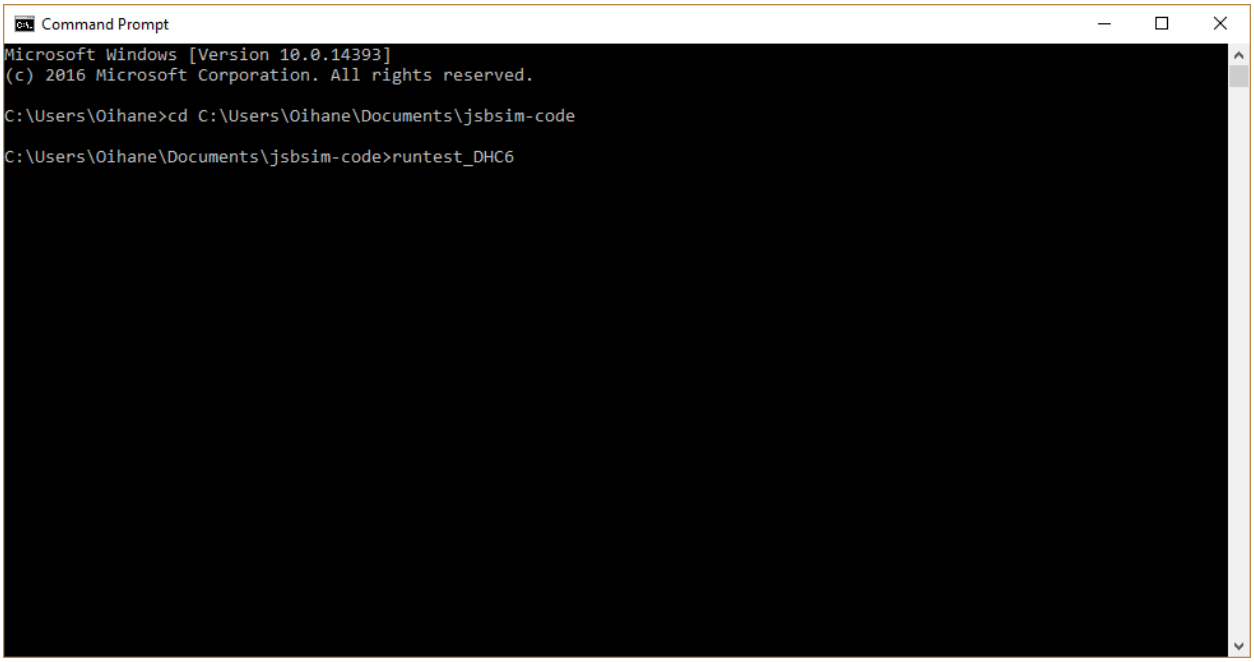


Figure 3. JSBSim program command line in batch mode

```
rem Remove the old results file
del /Q aircraft\DHC6\test\TestDHC6_Out.csv

rem Run the test
Debug\JSBSim --script=aircraft\DHC6\test\DHC6-test.xml

rem Copy the csv file to another location and change its name
copy JSBoutDHC6.csv aircraft\DHC6\test\
ren aircraft\DHC6\test\JSBoutDHC6.csv TestDHC6_Out.csv

rem Generate gnuplot to the screen
gnuplot aircraft\DHC6\test\PlotOrbitDHC6.p
```

1.1.1. Script Definition

The simulation task is defined in the script alongside the aircraft and its initial state. Events activate when a condition (declared within JSBSim) is met and a series of actions are activated. Each event is triggered once unless the condition associated with it is declared "continuous" or "persistent", making it constantly evaluated during the simulation.

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl"
href="http://jsbsim.sourceforge.net/JSBSimScript.xsl"?>
<runscript xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="http://jsbsim.sf.net/JSBSimScript.xsd"
    name="C172 cruise at 4K, 100% power">
  <description>This run is for testing the C172 altitude hold autopilot and
cruise performance</description>
  <use aircraft="c172x" initialize="reset01"/>

  <run start="0.0" end="50" dt="0.0083333">

    <property value="0"> simulation/run_id </property>

    <event name="Hold heading and altitude">
      <condition>simulation/sim-time-sec ge 5</condition>
      <set name="ap/heading_setpoint" value="200"/>
      <set name="ap/heading_hold" value="1"/>
      <set name="ap/altitude_setpoint" value="4000.0"/>
      <set name="ap/altitude_hold" value="1"/>
      <notify>
        <property> attitude/psi-rad </property>
        <property> attitude/theta-rad </property>
        <property> attitude/phi-rad </property>
        <property> position/h-agl-ft </property>
      </notify>
    </event>

  </run>

</runscript>
```

In the example framed above extracted from a script demo, the Cessna 172 aircraft is tested for the autopilot hold and cruise performance. The aircraft and its initial state is declared with `<use aircraft=" " initialize=" "/>` and the simulation conditions are defined with `<run start=" " end=" " dt=" ">`. The event extracted from the script shows that a series of autopilot properties get updated 5 seconds after the simulation starts. With the `</notify>` function, the Euler angles and the altitude are displayed on the command window when the event is triggered.

1.1.2. Visualization in FlightGear

By default, the standalone mode does not allow for visualization. With an extra line of code in the aircraft configuration file, the model can share its output with FlightGear, enabling the visual performance. For a successful exchange of properties, the same input/output information must be configured in FlightGear, since JSBSim runs as an external model.

Assume that following line of code is declared in the JSBSim in aircraft configuration file:

```
  <output name ="localhost" type="FLIGHTGEAR" port="5500" protocol="UDP"
rate="10"> </output>
```

Then, the FDM in FlightGear must be selected as external and the input property in FlightGear must be:

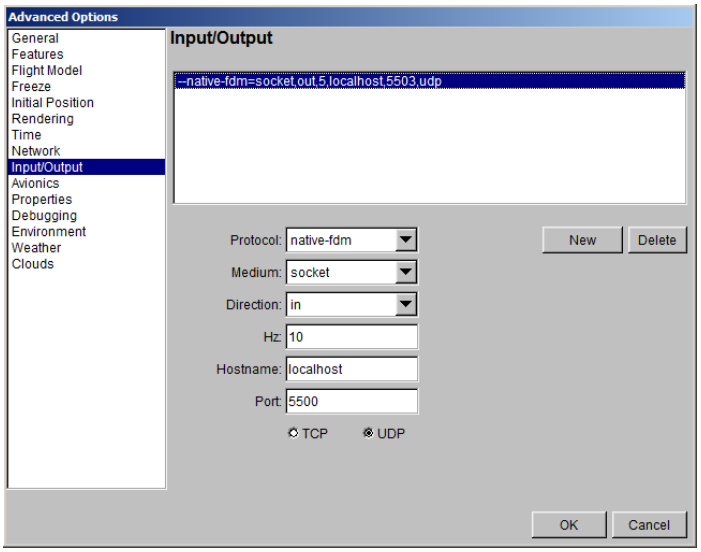


Figure 4. FlightGear interface. Advanced options. Input/output properties

The visualization of the performance will start as soon as the simulation launches. It should be noted that the visualization will depend on the JSBSim deltatime unless otherwise indicated.

1.2. Integration in FlightGear

The second way to run an aircraft designed in JSBSim is by integrating it directly into FlightGear [9]. The main difference between this mode and the mode explained in Section 1.1.2. is that in this case, only the aircraft configuration and its related files are needed since FlightGear holds the simulation. This is extremely useful when the aircraft is flown manually with a controller, whereas it becomes impractical when choosing a pre-defined task.

For a better understanding, the Figure 5 shows a simplified structure of FlightGear formed by aircraft (expressed in JSBSim FDM), Air Traffic Control (ATC), airport, scenery and other models:

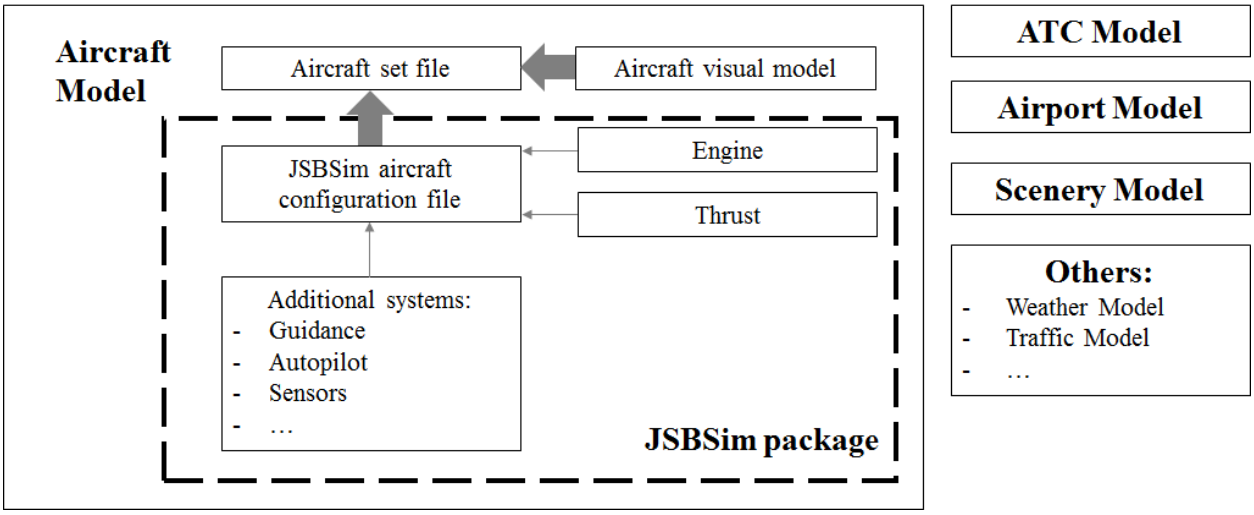


Figure 5. FlightGear block structure

Copy the required files in the source folder, including the aircraft information, into FlightGear to run the simulation. If everything has been done correctly, the selected aircraft will be listed as available. Before launching the simulation, ensure that the flight model option “jbs” is selected in Advanced Options (Figure 6). The model will start on the ground or in the air and can be initiated and controlled by a flight controller, a joystick, or with the computer keyboard.

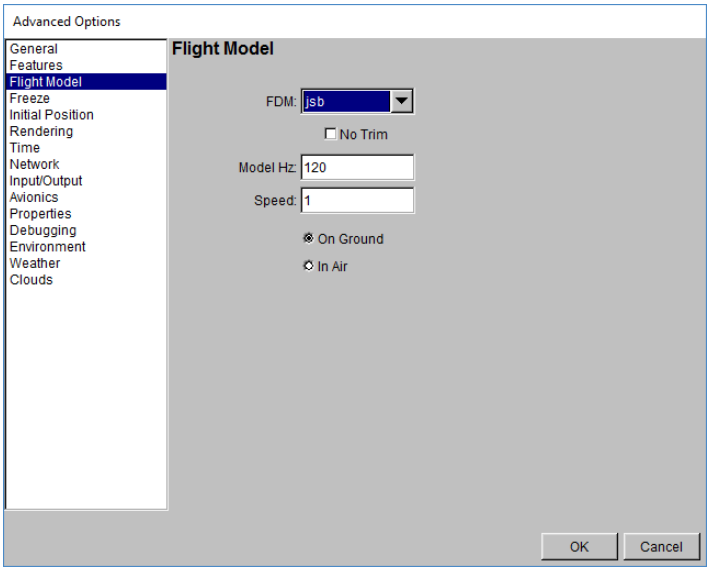


Figure 6. FlightGear interface. Advanced options. Flight Model

Note that in this mode there is no need to add any shared properties in the aircraft configuration file, since FlightGear will be generating the output straight from the simulation.

2. Classes, Class Hierarchy and Model Class

The JSBSim complete package includes all the source codes for a basic simulation; however, the user can create new systems depending on the required task. These systems might include autopilot, Guidance, Navigation, and Control (GNC) and specific onboard instruments. JSBSim is designed to be modified depending on the vehicle and user's needs. C++ represents an excellent programming language to cover all the requirements in JSBSim; it provides the required management tool for extracting, calculating and propagating data between classes as part of its object-oriented programming.

The JSBSim source code structure works like any other computer model in C++. The location and dynamics of the aircraft are given by the mathematical expressions [10]. The package models the metrics, the computation of the forces and moments, and the propagation/output of the dynamics, among others. Likewise, JSBSim also needs basic mathematical elements such as functions, tables, and quaternions to handle mathematical transformations.

The code is distributed in classes depending on their function. The collection of described high-level classes could be classified in:

- The **executive class** *FGFDMExec* initializes and runs the package from the application calling JSBSim. The simulation starts and finishes according to the script that includes the simulation details. When *FGFDMExec* class is initialized, it creates model objects that define the aircraft according to the aircraft configuration file defined by the user.
- The **model class** *FGModel* represents the real physical classes and elements in the aircraft. The model classes are executed in order including *FGAerodynamics, FGAircraft, FGAtmosphere*, etc.
- The **math classes** (*FGColumnVector3, FGQuaternion, FGTable*, etc.) contain the mathematical operations needed to solve equations, transformations, and other relations in JSBSim.
- **I/O and initialization classes** (*FGInputSocket, FGOutputTextFile*, etc.) handle inputs and outputs to the system.
- The **basic classes** (*FGJSBBase* and *FGState*) provide common capabilities to all the classes such as message handling.

Focusing on the UAV computer model design, only the Model classes differ from the general aviation aircraft case; the remaining classes are necessary, and include the operators and simulation characteristics required for the correct implementation of JSBSim. The model classes inherited from *FGModel* are listed below, including a brief description and special conditions (if applicable) for the UAV case:

- The input class (*FGInput*)[1] manages the inputs to the model with `<input>` elements in the aircraft configuration file. When the software reads `<input>`, a communication between classes is open and an appropriate action taken.
- The atmosphere class (*FGAtmosphere*)[1] models the 1976 standard atmosphere including winds, turbulence and giving the values of pressure, density and temperature, depending on the location of the aircraft. Beware that the UAV case will only consider low altitude (under 1000ft)[2] and therefore, the model may be simplified to consider only the troposphere conditions. However, the author recommends keeping this model as it is for possible future uses.
- The FCS class (*FGFCS*)[1] manages a collection of flight control classes defined by the aircraft components (surface control elements, throttle, autopilot). In a simple UAV

[1]This class is required in all types of aircraft and fixed-wing UAVs.

[2]This value might differ with the upcoming Transport Canada regulations to be implemented in 2019.

simulation, the primary controls for the model are the surface command elements of the aircraft which include the ailerons, elevator, and rudder.

- The propulsion class (*FGPropulsion*)[1] manages from 0 to n number of engines (*FGEngine*). JSBSim includes different kinds of propulsion systems depending on the engines used to generate thrust. They include a piston engine model (*FGPiston* → *FGEngine*), a jet turbine engine model (*FGTurbine* → *FGEngine*), a turboprop engine model (*FGTurboProp* → *FGEngine*), a rocket engine model (*FGRocket* → *FGEngine*) and an electric engine model (*FGElectric* → *FGEngine*). However, only the piston and electric models are considered in the case of UAVs. The thrust generation (*FGThruster* → *FGForce*) presents the same scenario where among the options found in JSBSim – direct, nozzle (*FGNozzle* → *FGThruster*), propeller (*FGPropeller* → *FGThruster*) and rotor (*FGRotor* → *FGThruster*)– only the propeller is used in the fixed-wing UAV case. The propulsion system, including the engine and the origin of the thrust generation, are called from the aircraft configuration file.
- The mass balance class (*FGMassBalance*)[1] calculates the moments of inertia, Center of Gravity (CG), and mass over time. At initialization, the `<mass_balance>` section in the aircraft configuration file is read and for each sample time, the CG and mass are updated. This class is relevant in the UAV case when the aircraft is carrying a piston engine since the fuel consumption will highly update those values.
- The aerodynamics class (*FGAerodynamics*)[1] is a collection of manager classes with individual force and moment definitions. When `<aerodynamics>` is called in the aircraft configuration file, this class handles the corresponding aerodynamic calculations obtaining the forces and moments calculated for each of the axes.
- The inertial class (*FGInertial*)[1] initializes the radius and reference acceleration values.
- The ground reactions class (*FGGroundReactions*) models the ground reactions defined in the aircraft configuration file as `<ground_reactions>`. The two types of contacts are BOGEY, which is directly related to the landing gear and its contacts, and STRUCTURE, which is used to locate any aircraft contact type that is not part of the landing gear (wing tips, nose and tail). JSBSim models the landing gear set as a spring/damper model with `<spring_coeff>` and `<damping_coeff>`. It also models retractable landing gear for contacts but it is not applicable to most UAVs as their landing gear is non-retractable.

- The aircraft class (*FGAircraft*)[1] gathers all systems together; it initializes the aircraft model loading its properties with `<metrics>`, and obtains the contribution of each of the systems in the generation of forces and moments.
- The propagate class (*FGPropagate*)[1] models the equations of motion, giving the state of the vehicle from the forces and moments generated during flight.
- The auxiliary class (*FGAuxiliary*) models pilot sensed accelerations and other auxiliary parameters used for acceleration calculations in inertial space. This class is only required in case the UAV carries motion-based sensors onboard.
- The output class (*FGOutput*)[1] handles the simulation output. The desired output is defined in the aircraft configuration file. The classes generated include: CSV (datalog in csv), SOCKET (data sent to a socket output defined by an IP address), FLIGHTGEAR (socket to FlightGear) and TABULAR (columnar data).

Figure 7 expresses the JSBSim FGModel in operation; when JSBSim is initialized, FGFDMExec is executed generating all the models' objects that will be uploaded with the information contained in the aircraft configuration file. This allows the assembly of the 6-DoF aircraft computer model.

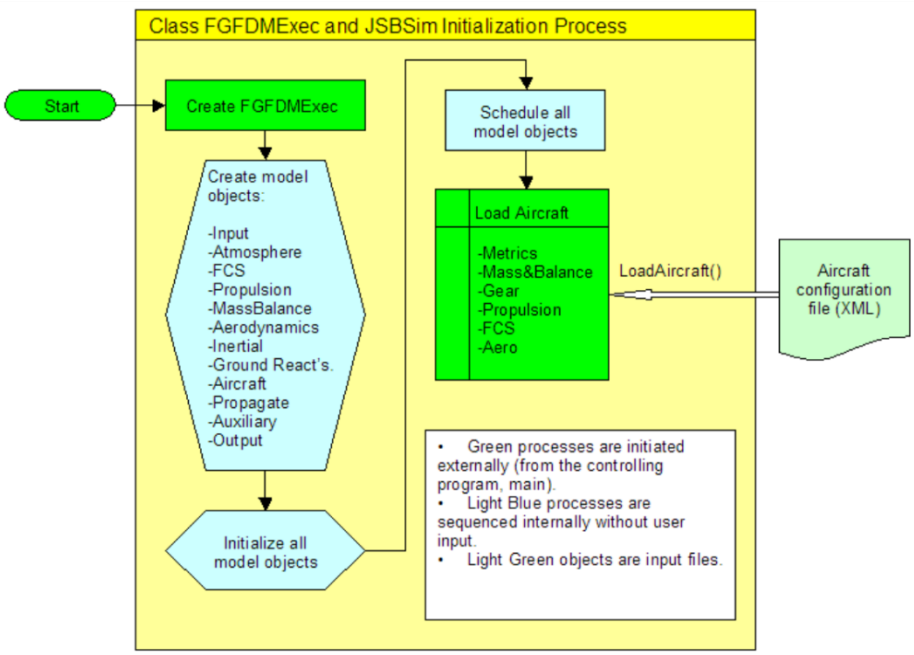


Figure 7. FGFDMExec and JSBSim Initialization process [1]

For a full collection of JSBSim classes, see reference [11].

3. Additional Features: Multiplayer Mode

An additional feature in FlightGear allows for the visualization of several independent models in one simulation by communicating with each other. The setup is similar to the FlightGear visualization in the JSBSim standalone mode (Section 1.1.2) except that the properties shared between one and another must be opposite.

For the multiplayer mode setup instructions (extracted from [12]), assume that one model is called “aircraft 1” (A1) and another “aircraft 2” (A2). For this mode, the IP of both computers is required to enable the communication; use the ipconfig command to retrieve that information.

In A1 follow these steps:

1- Launch FlightGear selecting the aircraft and the airport.
2- Select AI Models and Random Objects.
3- Select Multiplayer mode:
   - Callsign: GS_TEST1
   - Hostname: IP of A1
   - In: 5510
   - Out: 5520
4- Select Advanced options.
5- Flight model. Select the most appropriate model depending on the run mode (jsb, external, etc.).
6- Input/Output: Include the following variables:
   --native-fdm=socket, in, 10, localhost, 5500, udp (external FDM – JSBSim output)
   --native-ctrls=socket, out, 5, localhost, 5501, udp
   --native-fdm=socket, out, 5, IP of C2, 5504, udp (communication to A2)
7- Return and Run.

In A2, do the following:

1- Launch FlightGear, selecting the appropriate aircraft and the same airport as in A1.
2- Select AI Models and Random Objects.
3- In Multiplayer mode:
   - Callsign: GS_TEST2
   - Hostname: IP of A1

- In: 5520
- Out: 5510

4- Select Advanced Options.

5- Flight model: jsb, external, etc.

6- Input/Output: Add the following:

--native-fdm=socket, in, 10, localhost, 5502, udp (external FDM – JSBSim output)

--native-ctrls=socket, out, 5, localhost, 5503, udp

7- Return and Run.

Both aircraft should be displayed on the screen of A1 from the perspective of A1 (Figure 8). This particular Multiplayer mode allows one computer (A1) to be the host of the online simulation, while A2 only simulates the performance of the second aircraft.

Figure 8. Multiplayer mode in FlightGear with a Cessna 172 as seen from the cockpit of another aircraft

4. Case Study: Giant Big Stik Fixed-wing UAV

The Giant Big Stik R/C UAV (Figure 9) [13] is an aerobatic sport-scale aircraft belonging to the "Stik" series, manufactured by GreatPlanes [14]. It is powered by a fuel engine (Zenoah 26A) and flies like a full-size airplane. This model airplane has been widely used in the Remote Aerial Vehicles for ENvironment-monitoring (RAVEN) group at Memorial University of Newfoundland in St. John's, Newfoundland and Labrador (NL), Canada, for many years. Its aerobatic characteristics are the most significant highlight of this model.

Figure 9. Giant Big Stik on the field before tests

4.1. Giant Big Stik Aircraft Modelling

Based on the package structure described in Figure 1, the minimum set of blocks are created as part of the aircraft modelling:

- B3-initialization file: the simulation is held near Clarenville, NL, Canada with a certain initial UAV airspeed and no wind.

```
<initialize name="ini03">
  <vc unit="KTS"> 38.87689112646 </vc>
  <longitude unit="DEG"> -53.91752033  </longitude>
  <latitude unit="DEG">   48.2778129  </latitude>
  <phi unit="DEG">          0.0  </phi> <!-- Roll -->
  <theta unit="DEG">        0.0  </theta> <!-- Pitch -->
  <psi unit="DEG">        0.0  </psi> <!-- Yaw -->
  <altitude unit="FT">     984.252  </altitude>
  <hwind> 0.0 </hwind>
</initialize>
```

- B4-aircraft configuration file. It includes all the coefficients and parameters specific to the Giant Big Stik UAV, which are introduced in [15]. For this example, two cases are considered: an output file and a visualization while the simulation is running.

```
<fdm_config name="GBS" version="2.0" release="ALPHA"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:noNamespaceSchemaLocation="http://jsbsim.sourceforge.net/JSBSim.xsd">

    <fileheader>
        <author> Oihane Cereceda </author>
        <filecreationdate> 2015-11-11 </filecreationdate>
        <version> v4 </version>
        <description> Giant Big Stik JSBSim model
        v1: Files and values created from Aeromatic v0.82
        v2: Files modified according to more specific data
        v3: Validate model using scripts and plotting data </description>
        v4: Output to FlightGear to visualize the performance
    </fileheader>

    <metrics>

    <mass_balance>

    <ground_reactions>

    <propulsion>

    <flight_control name="FCS: unnamed">

    <aerodynamics>

  <output name ="localhost" type="FLIGHTGEAR" port="5500" protocol="UDP"
rate="10"> </output>

  <output name="GBS_Out.csv" type="CSV" rate="100">

</fdm_config>
```

4.2. Giant Big Stik in Standalone Mode

As an example of the use of JSBSim with UAV applications, a simple task is defined where the model turns left using the ailerons at maximum deflection in open-loop. In this example, which has been used in previous work [16] for validation purposes, the aileron is set to its maximum value in 5 seconds and returns back to 0.0 in 6.5 seconds.

```
  <use aircraft="GBS" initialize="ini03"/>
  <run start="0" end="35" dt="0.01">

    <event name="Set engine throttle">
```

```
    <event name="Set aileron max. Turn left">
      <condition> simulation/sim-time-sec ge 5.0 </condition>
      <set name="fcs/aileron-cmd-norm" action="FG_STEP" value="-0.569"
tc="1"/>
      <notify/>
    </event>

    <event name="Set aileron to zero">
      <condition> simulation/sim-time-sec ge 6.5 </condition>
      <set name="fcs/aileron-cmd-norm" action="FG_STEP" value="0.0"
tc="1"/>
      <notify/>
    </event>
```

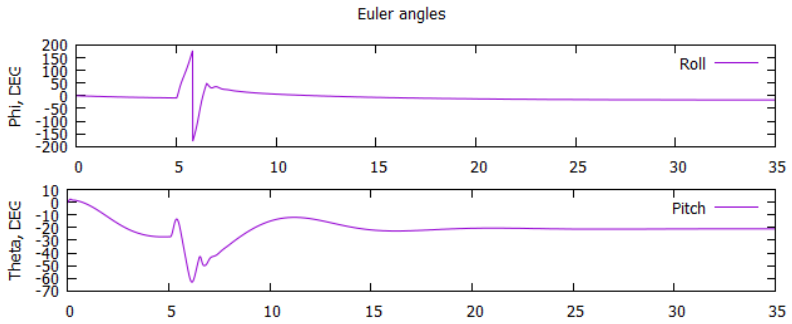


Figure 10. UAV roll and pitch angles

The simulation starts with the UAV in the air, meaning that it needs a few seconds to reach a stable situation before any command is sent to the model. This is also reflected when, in the case of real flight, the R/C model switches from manual to unmanned mode. The simulation test shown here expresses a special case where the ailerons are shifted to their maximum value in order to evaluate the dynamics of the system when performing an extreme manoeuvre. Figure 10 (plots generated using Gnuplot from JSBSim datalog output) shows how the roll angle is affected by changes in the aileron deflection; the aircraft quickly spins around and stabilizes as soon as the ailerons are back to 0.0.

The coherence of the system is noticeable. When the aircraft rolls, the lift vertical component is no longer balanced and the weight creates a loss in altitude and pitch angle. The response is presented in Figure 11 where FlightGear has been used to visualize the performance of the aircraft during the simulation.

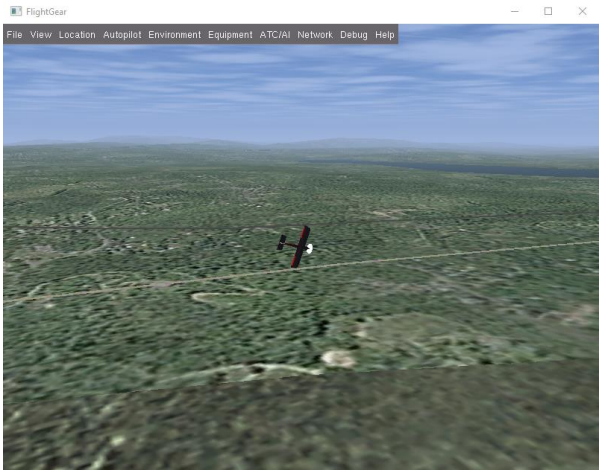
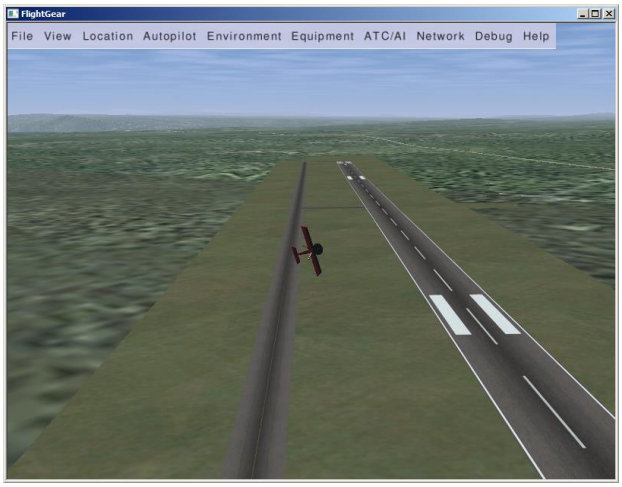


Figure 11. Giant Big Stik in FlightGear v2.0.0 performing an aerobatic manoeuvre in JSBSim standalone mode

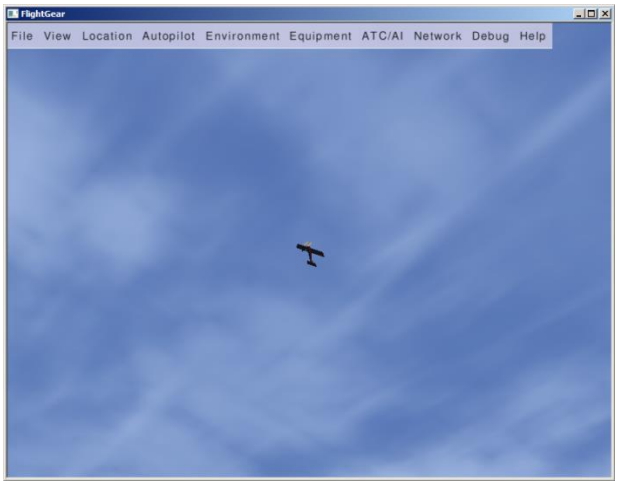
4.3. Giant Big Stik Integrated into FlightGear

This JSBSim feature allows the user to manually fly the model using the FlightGear interface. Following the setup described in Section 1.2, and using the Futaba Interlink Elite Controller to fly the computer model, the simulation views are the following:

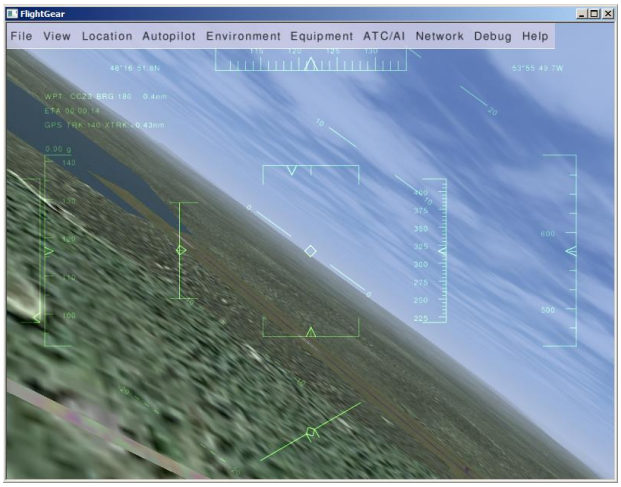


(a) Chase view of a turning manoeuvre

(b) Straight flight over the runway, chase view



(c) Fly-by view simulating what an R/C pilot would see from the ground



(d) Cockpit view during flight

Figure 12. Giant Big Stik manual flight in FlightGear v2.0.0

References

[1] J. S. Berndt and JSBSim Development Team, “JSBSim, An open source, platform-independent, flight dynamics model in C++.” 2011.

[2] Universita degli Studi di Napoli Federico II, “ADAG | Aircraft Design & AeroFlightDynamics Group.” [Online]. Available: http://www.adag.unina.it/english/index.html. [Accessed: 10-Oct-2017].

[3] Y. Zhang and S. Mcgovern, “Mathematical Models for Human Pilot Maneuvers in Aircraft Flight Simulation,” in *ASME 2009 International Mechanical Engineering Congress and Exposition*, 2009.

[4] “FlightGear Flight Simulator.” [Online]. Available: http://www.flightgear.org/. [Accessed: 28-Apr-2017].

[5] JSBSim Development Team, “JSBSim Flight Dynamics Model download.” [Online]. Available: https://sourceforge.net/projects/jsbsim/. [Accessed: 20-Sep-2018].

[6] JSBSim Development Team, “JSBSim Open Source Flight Dynamics Model.” [Online]. Available: http://jsbsim.sourceforge.net/. [Accessed: 09-Oct-2017].

[7] “OpenEaagles.” [Online]. Available: http://www.openeaagles.org/wiki/doku.php?id=start. [Accessed: 19-Oct-2017].

[8] “Gnuplot.” [Online]. Available: http://www.gnuplot.info/. [Accessed: 20-Sep-2018].

[9] “JSBSim - FlightGear wiki.” [Online]. Available: http://wiki.flightgear.org/JSBSim. [Accessed: 21-Apr-2017].

[10] M.V. Cook, *Flight Dynamics Principle*, 2nd ed. 2007.

[11] JSBSim Development Team, “JSBSim Flight Dynamics Model: JSBSim.” [Online]. Available: http://jsbsim.sourceforge.net/JSBSim/. [Accessed: 09-Oct-2017].

[12] J. Stevenson, “Small UAV 4D Simulation in MATLAB/Simulink and FlightGear. User Description Document,” 2013.

[13] Great Planes, “Giant Big Stik ARF Instruction Manual,” no. 217, pp. 13–15, 2005.

[14] M. M. C. Great Planes, “Great Planes Giant Big Stik ARF.” [Online]. Available: http://www.greatplanes.com/airplanes/gpma1224.php. [Accessed: 09-Oct-2017].

[15] J. D. Stevenson, “Assessment of the Equivalent Level of Safety Requirements for Small Unmanned Aerial Vehicles,” Memorial University of Newfoundland, 2015.

[16] O. Cereceda, L. Rolland, and S. O’Young, “Validation discussion of an Unmanned Aerial Vehicle (UAV) using JSBSim Flight Dynamics Model compared to MATLAB/Simulink AeroSim Blockset,” in *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2016, pp. 3989–3994.